

PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI

PPPPPPPP	LL	IIIIII	WW	WW	RRRRRRRR	IIIIII	TTTTTTTTTT	EEEEEEEEEE	
PPPPPPPP	LL	IIIIII	WW	WW	RRRRRRRR	IIIIII	TTTTTTTTTT	EEEEEEEEEE	
PP	PP	II	WW	WW	RR	RR	TT	EE	
PP	PP	II	WW	WW	RR	RR	TT	EE	
PP	PP	II	WW	WW	RR	RR	TT	EE	
PP	PP	II	WW	WW	RR	RR	TT	EE	
PPPPPPPP	LL	II	WW	WW	RRRRRRRR	II	TT	EEEEEEEE	
PPPPPPPP	LL	II	WW	WW	RRRRRRRR	II	TT	EEEEEEEE	
PP	LL	II	WW	WW	RR	RR	TT	EE	
PP	LL	II	WW	WW	RR	RR	TT	EE	
PP	LL	II	WWWW	WWWW	RR	RR	TT	EE	
PP	LL	II	WWWW	WWWW	RR	RR	TT	EE	
PP	LLLLLLLLLLLL	IIIIII	WW	WW	RR	RR	TT	EEEEEEEEEE	
PP	LLLLLLLLLLLL	IIIIII	WW	WW	RR	RR	TT	EEEEEEEEEE	

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLL IIIII
LLLLLLLLLL IIIII
SSSSSSSS
SSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSS
SSSSSSSS

```



```
0000 1      .title pli$write - pl1 runtime write file
0000 2      .ident /1-003/                               ; Edit WHM1003
0000 3
0000 4 :
0000 5 :*****
0000 6 :*
0000 7 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :*  ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :*  TRANSFERRED.
0000 17 :*
0000 18 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :*  CORPORATION.
0000 21 :*
0000 22 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28 :
0000 29 :
0000 30 :++
0000 31 : facility:
0000 32 :
0000 33 :     VAX/VMS PL1 runtime library.
0000 34 : abstract:
0000 35 :
0000 36 :     This module contains the pl1 runtime routine for writing a record to
0000 37 :     a file.
0000 38 :
0000 39 : author: c. spitz 18-jul-79
0000 40 :
0000 41 : modified:
0000 42 :
0000 43 :     Bill Matthews 22-Sep-81
0000 44 :
0000 45 :     Fix to get attributes of a close file from the declared
0000 46 :     attributes rather than the current attributes
0000 47 :
0000 48 :
0000 49 :     1-003  Bill Matthews  29-September-1982
0000 50 :
0000 51 :     Invoke macros $defdat and rtshare instead of $defopr and share.
0000 52 :
0000 53 :--
0000 54 :
0000 55 :
0000 56 :+
0000 57 : external definitions
```

```

0000 58 :-
0000 59
0000 60      $deffcb      ;define file control block offsets
0000 61      $defdat     ;define operand node data types
0000 62      $rabdef     ;define rab offsets
0000 63      $rmsdef     ;define rms error codes
0000 64      $fabdef     ;define fab offsets
0000 65
0000 66 ;+
0000 67 ; local definitions
0000 68 :-
0000 69 $offset 4,positive,<-      ;define arguments
0000 70      <fcbaddr,4>,-          ;addr of fcb
0000 71      <fromaddr,4>,-        ;addr of from
0000 72      <fromlen,2>,-         ;length of from
0000 73      <fromtyp,2>,-         ;data type of from
0000 74      <keyaddr,4>,-         ;addr of keyfrom
0000 75      <keylen,4>,-          ;length of keyfrom
0000 76      <keytyp,4>,-          ;data type of keyfrom
0000 77      <fxcaddr,4>,-         ;addr of fixed control
0000 78      <fxclen,2>,-          ;len of fixed control
0000 79      <fxctyp,2>,-          ;data type of fixed control
0000 80      <recidto,4>,-         ;addr of record id to
0000 81      >
0000 82
0000 83      rtshare
0000 84

```

;sharable


```
0000 86
0000 87 :++
0000 88 :pli$write -- write a record to a file
0000 89 :
0000 90 :functional description:
0000 91 :
0000 92 :This routine writes a record to a pl1 file.
0000 93 :
0000 94 :inputs:
0000 95 :    (ap) = number of arguments
0000 96 :           3 if no key, no options
0000 97 :           6 if key, no options
0000 98 :           9 if any options
0000 99 :    4(ap) = address of the fcb for this file
0000 100 :    8(ap) = address of the from reference
0000 101 :   12(ap) = word containing the length of the from reference
0000 102 :   14(ap) = word containing the data type of the from reference
0000 103 :   16(ap) = address of the key reference
0000 104 :   20(ap) = size/prec of key
0000 105 :   24(ap) = data type of key
0000 106 :   28(ap) = addr of fixed control
0000 107 :   32(ap) = word containing length of fixed control
0000 108 :   34(ap) = word containing data type of fixed control
0000 109 :   36(ap) = addr of record id to
0000 110 :
0000 111 :outputs:
0000 112 :    fcb_l_attr
0000 113 :    atr_m_delete and atr_m_virgin are set false
0000 114 :    atr_m_write and atr_m_currec are set true
0000 115 :    fcb_q_rfa is set to the rfa of the record that was written
0000 116 :
0000 117 :side effects:
0000 118 :    the record is written to the file.
0000 119 :
0000 120 :--
0000 121 :
0000 122 :.entry pli$write,^m<r2,r3,r4,r5,r6,r7,r8>
0002 123 :
0002 124 :check arguments. at least 3 arguments must be present.
0002 125 :
003  6C  D1 0002 126 :    cmpl    (ap),#3           ;enough arguments?
0005 127 :    beql    10$              ;if eql, yes
006  6C  D1 0007 128 :    cmpl    (ap),#6           ;enuf args?
000A 129 :    beql    10$              ;if eql, yes
009  6C  D1 000C 130 :    cmpl    (ap),#9           ;enuf args?
000F 131 :    beql    10$              ;if eql, yes
005  05  13 0011 132 :    clrl    r0               ;indicate not enough parms
00140 133 :    brw     fail             ;and fail
52  04  AC  D0 0016 134 10$:    movl    fcbaddr(ap),r2      ;get address of fcb
53  0C  A2  D0 001A 135 :    movl    fcb_l_attr(r2),r3 ;get attributes
001E 136 :
001E 137 :open the file if necessary. the open attributes will be record and
001E 138 :if update was not specified in the file declaration, output. if the
001E 139 :file is not opened after calling open, an error is signaled.
001E 140 :
29 53  01  E0 001E 141 :    bbs     #atr_v_opened,r3,30$ ;if file not opened
00001000 8F  DD 0022 142 :    pushl   #atr_m_record      ;then request record
```



```
03 10 A2 04 E0 0028 143      bbs      #atr_v_update, -  
                                fcb_l_attr(r2),20$      ;if update not specified  
                                #atr_m_output,(sp)      ;then request output  
                                r2                    ;push address of fcb  
                                calls #2,g^pli$open      ;open the file  
                                movl fcb_l_attr(r2),r3    ;get the new attributes  
                                bbs #atr_v_opened,r3,30$  ;if file still not opened  
                                movl #pli$_open,r0       ;then set open failure  
                                brw fail                ;and fail  
                                004B 152  
                                004B 153 ; check file attributes. file must have record. file must not have  
                                004B 154 ; input. if file has update, it must have direct.  
                                004B 155  
50 0A 53 0C E0 004B 156 30$: bbs #atr_v_record,r3,40$ ;if file does not have record  
00000000'8F D0 004F 157      movl #pli$_notrec,r0      ;then set not record file  
00FD 31 0056 158      brw fail                        ;and fail  
0A 53 06 E1 0059 159 40$: bbs #atr_v_input,r3,50$      ;if file has input  
00000000'8F D0 005D 160      movl #pli$_writein,r0     ;then set attempted write to input file  
00EF 31 0064 161      brw fail                        ;and fail  
                                0067 162  
                                0067 163 ; process from reference. copy size and address of from to rab.  
                                0067 164  
28 54 62 A2 9E 0067 165 50$: movab fcb_b_rab(r2),r4      ;get address of rab  
0A 08 AC D0 006B 166      movl fromaddr(ap),rab$l_rbf(r4) ;copy address of buffer  
0C AC DD 0070 167      pushl fromlen(ap)                ;push size and data type  
00000000'GF 01 FB 0073 168      calls #1,g^pli$$bytesize ;get byte size  
03 50 E8 007A 169      blbs r0,55$                    ;if invalid data type  
00D6 31 007D 170      brw fail                        ;then fail  
22 A4 51 B0 0080 171 55$: movw r1,rab$w_rsz(r4)         ;copy size of buffer  
0E AC 0B B1 0084 172      cmpw #dat_k_char_var,fromtyp(ap) ;if writing from char var  
13 12 0088 173      bneq 56$                          ;then  
0B 53 0D E0 008A 174      bbs #atr_v_scalar,r3,551$    ;if scalar varying, write it all  
22 A4 08 BC B0 008E 175      movw @fromaddr(ap),rab$w_rsz(r4) ;copy current size to rab  
28 A4 02 C0 0093 176      addl #2,rab$l_rbf(r4)         ;bump address so we don't write length  
04 11 0097 177      brb 56$                          ;cont  
22 A4 02 A0 0099 178 551$: addw #2,rab$w_rsz(r4)        ;include vcha len in write size  
                                009D 179  
                                009D 180 ; process options  
                                009D 181  
2C A4 D4 009D 182 56$: clrl rab$l_rhb(r4)              ;assume no fixed control from  
09 6C D1 00A0 183      cmpl (ap),#9                    ;options specified?  
14 12 00A3 184      bneq 59$                          ;if neg, no  
50 1C AC 9E 00A5 185      movab fxcaddr(ap),r0          ;get addr of fixed control  
00000000'GF 16 00A9 186      jsb g^pli$$fxdctlfrom      ;process fixed control  
50 24 AC D0 00AF 187      movl recidto(ap),r0          ;get addr of record id to  
00000000'GF 16 00B3 188      jsb g^pli$$valrecidto      ;validate record id to  
                                00B9 189  
                                00B9 190 ; process keyfrom option. if keyfrom is specified, the file must be keyed.  
                                00B9 191 ; the keys size and address is copied to the rab. keyed access is set in  
                                00B9 192 ; the rab.  
                                00B9 193  
06 6C D1 00B9 194 59$: cmpl (ap),#6                    ;keyfrom specified?  
23 19 00BC 195      blss 70$                          ;if lss, no  
50 10 AC D0 00BE 196      movl keyaddr(ap),r0          ;copy address of key  
1D 13 00C2 197      beql 70$                          ;if addr = 0 then no keyfrom  
0A 53 08 E0 00C4 198      bbs #atr_v_keyed,r3,60$      ;if file is not keyed  
50 00000000'8F D0 00C8 199      movl #pli$_notkeyd,r0    ;then set not keyed file
```



```
0084 31 00CF 200      brw      fail      ;and fail
      35 A4 94 00D2 201 60$:      clrb      rab$b_krf(r4)      ;writes always go to primary index
      10 AC 9E 00D5 202      movab     keyaddr(ap),r0      ;point to key descr
00000000'GF 16 00D9 203      jsb      g^pli$$writekey_r8      ;process key
      29 11 00DF 204      brb      100$      ;continue
      00E1 205      ;
      00E1 206      ; sequential write, make sure file has seq. set sequential access in rab.
      00E1 207      ; if delete set, find the deleted record.
      00E1 208      ;
      0A 53 0A E0 00E1 209 70$:      bbs      #atr_v_seql,r3,80$      ;if not seq. file
50 00000000'8F D0 00E5 210      movl     #pli$_notseq,r0      ;set not sequential file
      0067 31 00EC 211      brw      fail      ;and fail
      0A 53 08 E1 00EF 212 80$:      bbc      #atr_v_keyed,r3,90$      ;if file has keyed
50 00000000'8F D0 00F3 213      movl     #pli$_nokey,r0      ;set no key specified
      0059 31 00FA 214      brw      fail      ;and fail
      1E A4 00 90 00FD 215 90$:      movb     #rab$c_seq,rab$b_rac(r4) ;set sequential access in rab
14 00BC C2 05 E1 0101 216      bbc      #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),105$ ;if block io
      38 A4 D4 0107 217      clrl     rab$l_bkt(r4)      ;set for seq. access
      010A 218      ;
      010A 219      ; put the record
      010A 220      ;
      0B 00BC C2 05 E1 010A 221 100$:      bbc      #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),105$ ;if block io
      09 11 0110 222      $write     r4      ;do a write
      011B 223      brb      107$      ;cont
      0A 50 E8 0124 224 105$:      $put     r4      ;put the record
50 00000000'8F D0 0127 225 107$:      blbs     r0,120$      ;if put failed
      0025 31 012E 226 110$:      movl     #pli$_rmsr,r0      ;then set error code in rab
      0131 227      brw      fail      ;and fail
      0131 228      ;
      0131 229      ; successful completion
      0131 230      ;
      0C A2 02080000 8F CA 0131 231 120$:      bicl     #<atr_m_delete!atr_m_virgin>, - ;clear delete, virgin
      0139 232      fcb_l_attr(r2)      ;in fcb
      0C A2 00140000 8F C8 0139 233      bisl     #<atr_m_write!atr_m_currec>, - ;set write as last oper and
      0141 234      fcb_l_attr(r2)      ;current record incorrect
      20 A2 10 A4 7D 0141 235      movq     rab$w_rfa(r4),fcb_q_rfa(r2) ;set correct current record in fcb
      09 6C D1 0146 236      cmpl     (ap),#9      ;options passed?
      0A 19 0149 237      blss     125$      ;if lss, no
      50 24 AC D0 014B 238      movl     recidto(ap),r0      ;get addr of recid to
      04 13 014F 239      beql     125$      ;if eql, not specified
      60 10 A4 7D 0151 240      movq     rab$w_rfa(r4),(r0)      ;set recid to
      04 0155 241 125$:      ret      ;return
      0156 242
```

```
50 00000000'8F 12 12 0156 244 fail: bneq 10$ ;if not enough parms
    01 52 D0 0158 245 movl #pli$_parm,r0 ;then set not enough parms
    01 6C D4 015F 246 clrl r2 ;assume no fcb specified
    1C 19 D1 0161 247 cmpl (ap),#1 ;if fcb specified
50 52 04 AC D0 0164 248 blss 30$ ;then
00000000'8F D1 0166 250 movl fcbaddr(ap),r2 ;get address of fcb
    0F 12 D1 016A 251 10$: cmpl #pli$_rmsr,r0 ;rms rab error code?
    03 6C D1 0171 252 bneq 30$ ;if neq, no, cont
    0A 13 D1 0173 253 cmpl (ap),#3 ;key specified?
    53 10 AC 9E 0176 254 beql 30$ ;if eql, no, cont
00000000'GF 52 DD 0178 255 movab keyaddr(ap),r3 ;set addr of key descr for onkey
    50 DD 017C 256 jsb g^pli$$chk_keycnd ;check for key condition
00000000'8F 50 DD 0182 257 30$: pushl r2 ;set fcb addr
    03 03 DD 0184 258 pushl r0 ;set error code
    04 FB DD 0186 259 40$: pushl #pli$_error ;set error condition
    04 04 FB 018C 260 calls #3,g^pli$io_error ;signal the condition
    04 04 0193 261 ret ;return
    0194 262
    0194 263
    0194 264 .end
```


PLI\$WRITE
Symbol table

- pl1 runtime write file

K 3

16-SEP-1984 02:27:51 VAX/VMS Macro V04-00
6-SEP-1984 11:40:25 [PLIRTL.SRC]PLIWRITE.MAR;1

Page 7
(1)

```

$$TMP1      = 00000001
$$TMP2      = 00000054
ATR_M_CURREC = 00040000
ATR_M_DELETE = 00080000
ATR_M_OUTPUT = 00000020
ATR_M_RECORD = 00001000
ATR_M_VIRGIN = 02000000
ATR_M_WRITE  = 00100000
ATR_V_INPUT  = 00000006
ATR_V_KEYED  = 00000008
ATR_V_OPENED = 00000001
ATR_V_RECORD = 0000000C
ATR_V_SCALVAR = 0000000D
ATR_V_SEQL   = 0000000A
ATR_V_UPDATE = 00000004
DAT_K_CHAR_VAR = 0000000B
DIR...      = 00000001
FAB$B_FAC    = 00000016
FAB$V_BIO    = 00000005
FAIL         = 00000156 R      02
FCBADDR      = 00000004
FCB_B_ENVIR  = 000001C2
FCB_B_ESA    = 0000012E
FCB_B_EXTRA  = 0000003D
FCB_B_FAB    = 000000A6
FCB_B_IDENT  = 00000040
FCB_B_IDENT_NAM = 00000042
FCB_B_NAM    = 000000F6
FCB_B_NUMKCBS = 0000003C
FCB_B_RAB    = 00000062
FCB_C_LEN    = 000001C2
FCB_C_STRLEN = 00000034
FCB_L_ATTR   = 0000000C
FCB_L_BUF    = 00000014
FCB_L_BUF_END = 00000018
FCB_L_BUF_PT = 0000001C
FCB_L_CNDADDR = 000001B2
FCB_L_CONDIT = 000001AE
FCB_L_DTTR   = 00000010
FCB_L_ERROR  = 00000008
FCB_L_KCB    = 00000038
FCB_L_NEXT   = 00000000
FCB_L_PREVIOUS = 00000004
FCB_L_PRN    = 00000034
FCB_Q_RFA    = 00000020
FCB_W_COLUMN = 0000002E
FCB_W_IDENT_LEN = 00000040
FCB_W_LINE   = 00000030
FCB_W_LINESIZE = 0000002A
FCB_W_PAGE   = 00000032
FCB_W_PAGESIZE = 0000002C
FCB_W_REVISION = 00000028
FROMADDR     = 00000008
FROMLEN      = 0000000C
FROMTYP      = 0000000E
FXCADDR      = 0000001C
FXCLEN       = 00000020

```

```

FXCTYP      00000022
KEYADDR      00000010
KEYLEN       00000014
KEYTYP       00000018
PLI$$BYTESIZE ***** X 02
PLI$$CHK_KEYCND ***** X 02
PLI$$FXDCTLFROM ***** X 02
PLI$$VALRECIDTO ***** X 02
PLI$$WRITEKEY_R8 ***** X 02
PLI$IO_ERROR ***** X 02
PLI$OPEN     ***** X 02
PLI$WRITE    00000000 RG 02
PLI$_ERROR   ***** X 02
PLI$NOKEY    ***** X 02
PLI$NOTKEYD  ***** X 02
PLI$NOTREC   ***** X 02
PLI$NOTSQL   ***** X 02
PLI$OPEN     ***** X 02
PLI$PARM     ***** X 02
PLI$RMSR     ***** X 02
PLI$WRITEIN  ***** X 02
RAB$B_KRF    = 00000035
RAB$B_RAC    = 0000001E
RAB$C_SEQ    = 00000000
RAB$L_BKT    = 00000038
RAB$L_RBF    = 00000028
RAB$L_RHB    = 0000002C
RAB$W_RFA    = 00000010
RAB$W_RSZ    = 00000022
RECIDTO      00000024
SIZ...       = 00000001
SYS$PUT      ***** GX 02
SYS$WRITE    ***** GX 02

```


+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS\$	000001C2 (450.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PLI\$CODE	00000194 (404.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.07	00:00:00.33
Command processing	79	00:00:00.52	00:00:01.53
Pass 1	199	00:00:07.30	00:00:15.58
Symbol table sort	0	00:00:00.79	00:00:01.89
Pass 2	54	00:00:01.35	00:00:02.89
Symbol table output	11	00:00:00.07	00:00:00.28
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	354	00:00:10.15	00:00:22.53

The working set limit was 900 pages.
39078 bytes (77 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 694 non-local and 22 local symbols.
264 source lines were read in Pass 1, producing 14 object records in Pass 2.
21 pages of virtual memory were used to define 18 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	5
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	15

789 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS\$:PLIWRITE/OBJ=OBJ\$:PLIWRITE MSRC\$:PLIWRITE/UPDATE=(ENH\$:PLIWRITE)+LIB\$:PLIRTMAC/L

0309 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY